Multi-Character Motion Retargeting for Large-Scale Transformations*

Maryam Naghizadeh and Darren Cosker

University of Bath, Bath, United Kingdom
{M.Naghizadeh,D.P.Cosker}@bath.ac.uk

Abstract. Unlike single-character motion retargeting, multi-character motion retargeting (MCMR) algorithms should be able to retarget each character's motion correcly while maintaining the interaction between them. Existing MCMR solutions mainly focus on small scale changes between interacting characters. However, many retargeting applications require large-scale transformations. In this paper, we propose a new algorithm for large-scale MCMR. We build on the idea of interaction meshes, which are structures representing the spatial relationship among characters. We introduce a new distance-based interaction mesh that embodies the relationship between characters more accurately by prioritizing local connections over global ones. We also introduce a stiffness weight for each skeletal joint in our mesh deformation term, which defines how undesirable it is for the interaction mesh to deform around that joint. This parameter increases the adaptability of our algorithm for large-scale transformations and reduces optimization time considerably. We compare the performance of our algorithm with current state-of-the-art MCMR solution for several motion sequences under four different scenarios. Our results show that our method not only improves the quality of retargeting, but also significantly reduces computation time.

Keywords: Motion Retargeting · Computer Animation · Character Interaction · Mesh Deformation · Joint Stiffness · Space-Time Optimization

1 Introduction

Single-character motion retargeting (SCMR) is "the problem of adapting an animated motion from one character to another" [4]. SCMR is a widely studied field and has many applications ranging from animation to robotics. However, many of these applications require more than one character to be retargeted together, e.g. fighting characters in an animation or collaborative robot tasks. SCMR is lacking when it comes to solving such problems since retargeting each character independently results in interaction loss among the characters. We will

* This research is co-funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 665992, and the UK's EPSRC Centre for Doctoral Training in Digital Entertainment (CDE), EP/L016540/1.

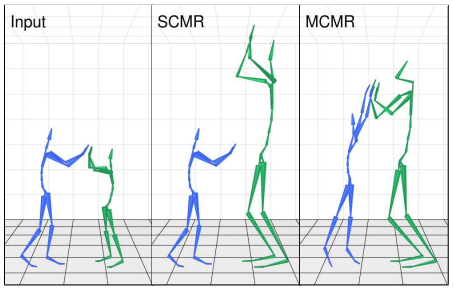


Fig. 1. The green character from the input (a) is retargeted to 2.0x its original size via SCMR (b) and MCMR (c). The interaction between blue and green characters is lost in SCMR while MCMR successfully preserves the high-five.

refer to these problems as multi-character motion retargeting (MCMR), which aims at generating motion for multiple target characters given the motion data for their corresponding source subjects. MCMR is different from individually retargeting multiple characters at the same time, which is illustrated in Fig. 1. When the green character from the input is individually retargeted to 2.0x its original size via SCMR, the high-five interaction between the blue and green characters is lost. However, this interaction is successfully maintained when an MCMR solution is used to perform the retargeting. MCMR is even more difficult when the scale change for one character is very different from the other one. Such cases are common in games and movies. For example, movies like *Avatar* and *Lord of the Rings* use motion retargeting to drive characters that are much taller or shorter than the human actors controlling them. Current solutions include using physical props to direct actor eye-lines towards the correct target above the other actors body, or having actors stand on boxes or stilts. Even so, a significant amount of clean-up is required, increasing post-processing time considerably. Moreover, acting under such circumstances is quite inconvenient for the actors.

In this paper, we propose an algorithm for large-scale MCMR that uses distance-based interaction meshes with anatomically-based stiffness weights. Existing MCMR solutions fail under large-scale character changes because they are tailored to cases where the height relationship is similar between source and target characters. These algorithms cannot adapt to increasing scale changes mostly because they do not consider human anatomy in their work and assume equal freedom of movement for all joints along the human skeleton. In order to attribute different levels of motion freedom for skeletal joints, we use stiffness weights, which have been successfully employed in large mesh deformations [13]. We use stiffness weights to define how costly it is for the interaction mesh to deform around each joint. The intuition behind stiffness weights is that they act like joint angle constraints; when a mesh does not deform around a joint after retargeting, the angles between its neighbors and itself do not change either. Our proposed method can successfully retarget motion for characters under large-scale transformations with greatly reduced computational time in comparison to existing MCMR approaches. To summarize, our contributions include:

1. Proposing the first MCMR algorithm tailored for large-scale character transformations.
2. Introducing an interaction mesh structure suitable for small and large-scale character transformations.
3. Introducing anatomically-based joint stiffness weights that improve retargeting quality while significantly reducing retargeting time.

2 Related Work

Single-character Motion Retargeting. Motion retargeting is defined as "the problem of adapting an animated motion from one character to another" [4]. Early retargeting solutions focused on retargeting motion between characters with identical skeletal structure [3, 4, 17], mainly using inverse kinematics and space-time optimization. More recent approaches include formulating motion retargeting as a distance-based optimization problem [2] and using recurrent neural networks [18].

Context-aware motion adaptation. Many retargeting studies do not consider the interaction of characters with the rest of the scene. However, retargeting characters independently often causes interaction artifacts in the scene and requires further manual work to fix them. Some studies define interaction descriptors between two objects, which are used to group objects according to their functionality or calculate the similarity between two scenes [8, 9, 14]. Zhao et al. introduce the *interaction bisector surface* (IBS) for capturing the spatial relationships among the objects in a scene [20]. IBS is mainly useful for applications with geometrically well-defined rigid objects and cannot be used to model interaction between deformable structures. These methods cannot readily be used for modelling the interaction space around a character due to its dynamic nature. Early solutions include defining constraints to achieve the desired relationship among the character and its surroundings [4, 12]. One recent solution is provided by Zhao et al., which extends the application of IBS [20] to character-object interaction [19].

Multi-character motion retargeting. Not much research has been conducted on maintaining inter-character relationship in motion retargeting. Ho et al. propose a method to maintain character interaction by introducing the concept of *interaction meshes* [7]. An interaction mesh is a structure that represents the spatial relationship among subjects and is obtained by performing *Delaunay tetrahedralization* [15] on the joint cloud of the subjects in the scene. The objective of their method is to solve the retargeting problem while preserving the structure of the interaction mesh. To do so, they formulate motion retargeting as a space-time optimization problem. The main limitation of this algorithm is that it does not always succeed in producing realistic postures especially when source and target characters have very different scales.

Ho et al. introduce a multi-resolution version of [7] by cutting the motion sequence into smaller pieces and solving each one separately, which makes the algorithm parallelizable and improves its computation time to some extent [6].

Another study uses the interaction mesh for interactive partner control [5]. In this study, the algorithm compares motion capture input to frames from a motion database that contains motion sequences of a duo (one labelled as active and the other as passive). The best match is decided via a metric that considers Euclidean distance from the input pose as well as temporal coherence to previous frames. The final motion is synthesized by using the method in [7] to retarget the active partner's pose to match the scale of the user.

Jin et al. propose the *Aura Mesh*, a volumetric mesh structure that encapsulates the spatial relationship between two skinned characters [10]. During the pre-processing stage, aura meshes are constructed for the source and target characters. At run-time stage, collision points of the aura meshes are used to embody the semantics of the interaction between characters. Inverse kinematics is used to generate the final motion by respecting the desired collision positions and minimizing a few other energy terms pertaining to motion naturalness such as pose preservation and foot contact. While very useful for retargeting close interaction between characters, this method suffers from a few disadvantages; retargeting is performed frame-by-frame and the output is interpolated to achieve smoothness, which can violate the original constraints. Furthermore, their method relies on having equal mesh topology for all characters in order to establish correspondence among aura meshes. This means that it cannot retarget characters with different mesh topologies. Finally, the method fails under large scale differences between source and target characters.

3 Methodology

First, we will formally define the inputs of the algorithm as depicted in Fig. 2.
Motion sequence ($\mathcal{V}$). Input motion contains 3D joint position data for source subjects over time. Let n be the number of frames in the motion sequence: $\mathcal{V} = (V_1, V_2, ...V_n)$. For each frame i, V_i is a vector composed of joint position vectors for all source subjects.
Joint stiffness matrix (S). Joint stiffness matrix is a new parameter that we introduce in this paper. It is a square diagonal matrix that contains stiffness weights for the joints:

$$S = \begin{pmatrix} s_1^1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & s_1^2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & s_1^3 & 0 & \cdots & 0 \\ 0 & 0 & 0 & s_2^1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & s_a^3 \end{pmatrix} \tag{1}$$

where a is the number of joints in the skeleton, and s_j^1, s_j^2 and s_j^3 hold the stiffness weight for the j-th joint along X, Y and Z axes, respectively. These weights define how undesirable it is for the interaction mesh to deform around the j-th joint along each axis.
Bone lengths (L). L is comprised of two vectors: L_S and L_T, which contain bone length values for source subjects and target characters.

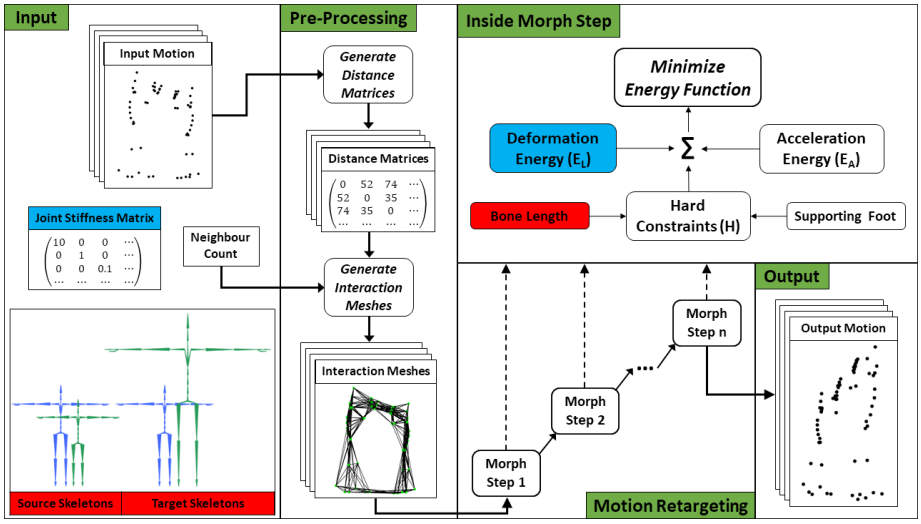


Fig. 2. An overview of our approach

Neighbor count (N). N is the number of connections per joint in the interaction mesh.

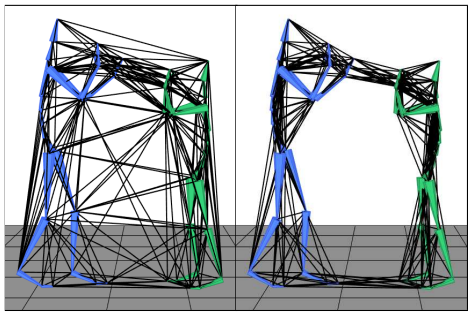
3.1 Pre-Processing

The pre-processing stage has two steps: calculating distance matrices and building interaction meshes.

Calculating distance matrices. The first step in pre-processing phase is calculating a distance matrix for each frame in the motion sequence, which contains the Euclidean distance between all joint pairs in the scene.

Distance-based interaction mesh. First introduced in [7], interaction mesh is a mesh structure that represents the spatial relationship between subjects for all frames along the motion sequence. The interaction mesh in [7] is obtained via Delaunay tetrahedralization of the 3D points that represent skeletal joints of the subjects in the frame. These joints constitute the vertices of the interaction mesh and a connection between a pair of vertices represents a spatial relationship between their corresponding joints. A sample interaction mesh generated via this method is depicted in Fig. 3(a).

We propose a new distance-based method to create interaction meshes. Given a neighbor count N, the algorithm builds an interaction mesh for each frame using its corresponding distance matrix to find the N closest joints in the space around each joint. By connecting each joint to its N closest joints, a distance-based interaction mesh is obtained that prioritizes local connectivity; unlike the interaction mesh constructed via Delaunay tetrahedralization. Fig. 3(b) shows a sample interaction mesh created using our distance-based approach where each joint has 12 connections. The intuition behind our distance-based approach comes from the fact that if interactions are preserved locally, they would be



(a) (b)

Fig. 3. Delaunay tetrahedralization produces interaction meshes with global connectivity (a), while our distance-based mesh prioritize local interactions (b).

maintained globally as well. Moreover, global connections can affect large-scale retargeting adversely because distant connections are often not meaningful or necessary for the spatial relationship between the characters. For instance, the connections between the elbow of the green character and the leg joints of the blue character in Fig. 3(a) force the green character to bend unnaturally when it is retargeted to a taller character.

3.2 Motion Retargeting with Joint Stiffness

Similar to [7], our algorithm takes on a step-wise approach to solve the retargeting problem. Instead of going directly from original scale to target scale, the algorithm divides the scale difference between source and target to k equal *morph steps* and solves them separately. First, step 1 is solved, then step 2 and so on; until the final result is produced in the last step. The output of each step provides the input for the next step. The following equation describes the objective function that each morph step is trying to minimize in our method:

$$\arg\min_{V_i^{'},\lambda_i} \sum_{i=1}^{n} E_L + w_\Delta E_A + \lambda_i^T (H_i V_i^{'} - h_i) \tag{2}$$

which is the sum of deformation (E_L) and acceleration (E_A) energies subject to the satisfaction of hard constraints ($\lambda_i^T (H_i V_i^{'} - h_i)$). In this equation, $V_i^{'}$ is the vector containing the updated joint positions for frame i, λ_i represents the Lagrange multipliers for frame i, n is the number of frames, w_Δ is a constant weight assigned for acceleration energy, H_i is the hard constraint Jacobian matrix for frame i and h_i is a vector that contains the hard constraint values for frame i. Note that unlike [7], we do not consider the soft constraints (collision and positional constraints) in our energy function.

Deformation energy (E_L) is the term that preserves the spatial relationship between the subjects by minimizing the deformation of the interaction meshes during retargeting. Equation (3) is used in [7] to minimize the Laplacian deformation energy [1] of the interaction meshes:

$$E_L(V_i^{'}) = \frac{1}{2} V_i^{'\mathsf{T}} M_i^{\mathsf{T}} M_i V_i^{'} - b_i^{\mathsf{T}} M_i V_i^{'} + \frac{1}{2} b_i^{\mathsf{T}} b_i \tag{3}$$

where $V_i^{'}$ contains the updated joint positions for frame i, b_i holds the original Laplacian coordinates for frame i and M_i is the Laplacian matrix.

The deformation energy term defined by Eq. (3) assumes that deformation cost is equal for all joints. An unintended side effect of this is that the optimization tries to preserve the relative height relationship among the subjects. This results in a squashing effect when one of the subjects is being retargeted to a taller character and stretching effect when it is being retargeted to a shorter one. This issue becomes bolder for large-scale character transformations since the height relationship among the subjects is required to change after retargeting. This issue happens because Eq. (3) is oblivious to the fact that not all joints in the human body have the same freedom of movement. For example, elbow and knee joints can move more freely compared to the spine joints. In order to resolve this problem, we introduce a weight for each joint in the deformation energy that defines how *undesirable* it is for this joint to violate its position in the interaction mesh. We do this by adding the stiffness weight matrix ($\mathbf{S}$) to Eq. (3) and obtaining the following new deformation term:

$$E_L(V_i^{'}) = \frac{1}{2}V_i^{'\mathsf{T}}M_i^{\mathsf{T}}\mathbf{S}M_iV_i^{'} - b_i^{\mathsf{T}}\mathbf{S}M_iV_i^{'} + \frac{1}{2}b_i^{\mathsf{T}}\mathbf{S}b_i \tag{4}$$

Joint stiffness matrix defines a stiffness weight for each joint in the skeleton. The matrix instructs the objective function to tolerate deformation around joints with low stiffness value and protect the mesh structure around joints with high stiffness value. It should be noted that the stiffness value for a joint is meaningless on its own, and that our joint stiffness weights operate in a relative way (for example, weights of 2 and 20 for a pair of joints act the same as weights of 4 and 40). Therefore, the relationship among the values defines how costly it is for the objective function when the mesh deforms around the joints with respect to each other. With that in mind, we divide the joints in the skeleton to three levels of priority: *stiff*, *normal* and *loose*. A good analogy is to think of stiff, normal and loose stiffness levels as hard, medium and soft joint angle constraints. We make the assumption that the joints along the spine (including the hip joint) are the the least free in their movement in human skeleton and should be in the stiff category. We also make the assumption that the elbow and knee joints are the most freely-moving joints in the human skeleton and include them in the loose category. Finally, we label the rest of the joints in the skeleton as normal.

Note that we calculate Laplacian coordinates in a similar fashion to [7] and that Laplacian weights are inversely proportional to the Euclidean distance between the vertices in the interaction mesh. Also, hard constraints and acceleration energy are included in our energy function in a similar manner to [7].

4 Results and Discussion

We tested our algorithm on the following motion sequences: high-five, handshake, kicking, dance and Ogoshi[1]. Our selection includes both simple and complex

[1] Video results are provided in the supplemental material.

types of interaction to cover a wide range of motions. We refer to simple interactions as motion instances where the spatial relationship among the subjects is partial, such as shaking hands. On the other hand, the spatial relationship in complex interactions (e.g. dancing) span the whole body.

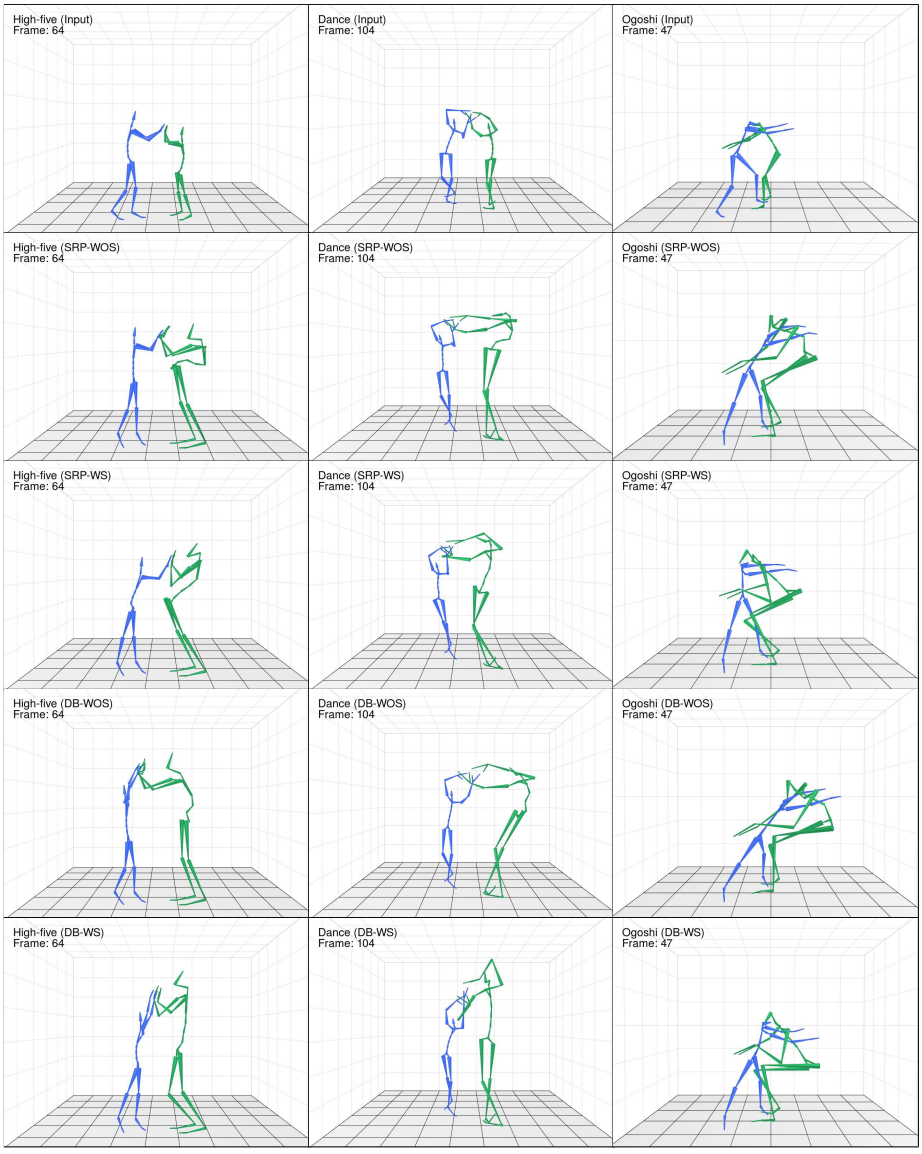
Setting parameter values. Unlike Delaunay tetrahedralization, our method receives the number of connections per joint as input. We tested our method's retargeting quality for various neighbor counts. For most interactions, 12 connections per joint provides satisfactory results. However, complex interactions may require up to 16 connections per joint. We set the stiffness weight for normal joints to 1. Therefore, for loose joints, the stiffness weight should be between 0 and 1, and for stiff joints, it can range from 1 to infinity. After experimenting with various weights, we observed that a too small stiffness value for loose joints creates unrealistic poses, and a very large stiffness weight for stiff joints increases the convergence time of the algorithm without producing any better results. We suggest that a good stiffness weight for loose joints is between 0.05 and 0.2, and a good stiffness weight for stiff joints lies between 5 and 50. For all our experiments, we used 0.1 and 10 for loose and stiff joints, respectively. Finally, as suggested by [7], w_{Δ} and the number of morph steps were set to 0.2 and 10 in all our experiments.

4.1 Retargeting Results

In order to analyze the effect of distance-based interaction meshes and joint stiffness on the final output separately, we conducted our experiments in four different scenarios: 1) Interaction mesh is generated via Delaunay tetrahedralization and the energy function does not include joint stiffness (SRP-WOS)[2]; 2) Interaction mesh is generated via Delaunay tetrahedralization and the energy function includes joint stiffness (SRP-WS); 3) Interaction mesh is generated via our distance-based method and the energy function does not include joint stiffness (DB-WOS); and 4) Interaction mesh is generated via our distance-based method and the energy function includes joint stiffness (DB-WS). We will use the terms "DB-WS" and "our method" interchangeably.

Figure 4 shows our results for three sample frames from high-five (Fig. 4(a)), dance (Fig. 4(b)) and Ogoshi (Fig. 4(c)) sequences. The second, third, fourth and fifth rows show the result of retargeting the green subject to 2.0x its original size using SRP-WOS. SRP-WS, DB-WOS and DB-WS, respectively. The height relationship between the subjects changes after doubling the green subject's scale. SRP-WOS fails at adapting to the new height relationship between the characters and tries to maintain the original relationship by squashing the green character along the spine. SRP-WS performs slightly better as the stiffness weights preserve the natural alignment of the spine joints. However, the global nature of the Delaunay mesh forces the green character to bend backwards unrealistically in the high-five sequence. DB-WOS manages to adapt to the new height relationship among the characters but the spine looks crooked. DB-WS successfully adapts to the new scale of the green character by bending the knee

[2] This is the same [7] with the exclusion of collision and positional constraints.



(a) (b) (c)

Fig. 4. The first row depicts three sample frames from high-five, dance and Ogoshi motion sequences. The second, third, fourth and fifth rows show the result of retargeting the green subject to 2.0x its original size using spatial relationship preserving method without (SRP-WOS) and with (SRP-WS) stiffness, and distance-based method without (DB-WOS) and with (DB-WS) stiffness, respectively. DB-WS produces realistic results by bending the knees and arms when needed to preserve the interaction between the characters without squashing the spine joints.

joints when required and keeping the spine joints intact. One can notice the difference between the effect of using distance-based meshes versus adding joint stiffness by comparing SRP-WS and DB-WOS. This example highlights how adding stiffness improves posture while using distance-based meshes improves the adaptability of the algorithm to changes in the height relationship among characters.

Another noticeable observation is that head orientation cannot be retargeted properly with any of the four methods. This is because the head orientation depends on the subject's gaze target. Since the gaze target can be anywhere in the space, retargeting the head orientation correctly is not possible solely using the interaction mesh. To resolve this issue, the gaze target must be detected during the pre-processing phase (e.g. using an eye tracker) and the direction of the head must be adjusted after retargeting to face the gaze target.

4.2 Computation Time

The algorithm is implemented in MATLAB 2017b and all the computation times are recorded on an Intel Xeon 2.90GHz CPU with 32GB RAM.

Table 1(a) compares pre-processing times for when Delaunay tetrahedralization is used for mesh generation versus when our distance-based method is used. The pre-processing times for the distance-based method is slightly longer than the Delaunay method, which is due to the time spent building distance matrices. Notice how pre-processing time increases for both scenarios with the number of joints per subject. This is expected as the time required for generating interaction meshes grows with the number of joints.

Table 1(b) compares retargeting time results for when joint stiffness is not included in the objective function versus when it is. We see that adding stiffness massively improves the retargeting time. This improvement happens because the joint stiffness matrix guides the optimizer towards the answer by inciting it to prioritize preserving the structure of the mesh around the joints with relatively higher stiffness values and not restrain the mesh around the joints with relatively lower values. Having different stiffness values for the joints makes the energy function an easier problem to solve since the deformation cost is not evenly distributed among the joints. This helps the optimizer to narrow down its search space much faster and find the solution in considerably less time. Another interesting observation is that without stiffness, the retargeting time for each frame increases with the number of frames, whereas this increase is more marginal with stiffness.

Table 1(c) compares the total run-time of our method versus the other three. While SRP-WS edges out DB-WS for best run-time, both methods have significant time advantage over the other two.

5 Conclusion and Future Work

In this paper, we introduced a new algorithm for large-scale multi-character motion retargeting (MCMR). We compared our method with the state-of-the-art MCMR solution. We demonstrated that our method not only outperforms

Table 1. Time analysis

a. Pre-processing time results[1]

Name	No. of frames	No. of joints[2]	Delaunay tetrahedralization		Distance-based	
			No. of connections	Pre-processing time[3]	No. of connections	Pre-processing time[3]
High-five	110	34	12.06	0.065	12	0.111
Handshake	110	34	12.28	0.066	12	0.112
Kicking	85	25	11.44	0.046	16	0.079
Dance	250	32	11.87	0.065	16	0.115
Ogoshi	255	25	11.59	0.054	12	0.082

b. Retargeting time results[1]

Name	No. of frames	No. of joints[2]	No. of connections	Retargeting time[3]	
				Without stiffness	With stiffness
High-five	110	34	12	7.95	0.30
Handshake	110	34	12	8.05	0.36
Kicking	85	25	16	2.39	0.24
Dance	250	32	16	31.27	0.78
Ogoshi	255	25	12	15.92	0.26

c. Total run-time results[1]

Name	No. of frames	No. of joints[2]	Total run-time[3]			
			SRP-WOS	DB-WOS	SRP-WS	DB-WS
High-five	110	34	7.90	8.06	0.37	0.41
Handshake	110	34	8.05	8.15	0.42	0.47
Kicking	85	25	2.40	2.46	0.30	0.32
Dance	250	32	31.32	31.38	0.84	0.89
Ogoshi	255	25	15.93	16.03	0.32	0.36

[1] The times in this table are obtained by averaging over 10 runs of the algorithm for high-five, handshake and kicking sequences and 5 runs for the dance and Ogoshi sequences. The standard deviation values are negligible in all cases. [2] per subject [3] sec/frame

existing skeletal MCMR approaches in terms of retargeting quality but it also improves run-time, is more scalable and can deal with larger sequences more efficiently. Our method produces animations that can be used to considerably reduce the amount of post-processing, by providing a much better starting point for further editing. One of the limitations of our work is that it does not include physics constraints such as balance. This can be resolved by adding a balance term to our energy function similar to [6]. Another direction for future work would be to include priors over the ranges of acceptable joint angles to avoid unlikely postures. Finally, it would be interesting to explore the effect of different forms of deformation energy such as Bi-Laplacian [11] and As-Rigid-As-Possible [16] on the quality of retargeting.

References

1. Alexa, M.: Differential coordinates for local mesh morphing and deformation. The Visual Computer **19**(2), 105–114 (May 2003)

2. Bernardin, A., Hoyet, L., Mucherino, A., Gonçalves, D., Multon, F.: Normalized euclidean distance matrices for human motion retargeting. In: Proceedings of the Tenth International Conference on Motion in Games. pp. 15:1–15:6. MIG '17, ACM, New York, NY, USA (2017)
3. Choi, K.J., Ko, H.S.: Online motion retargetting. The Journal of Visualization and Computer Animation **11**(5), 223–235 (2000)
4. Gleicher, M.: Retargetting Motion to New Characters. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques. pp. 33–42. SIGGRAPH '98, ACM, New York, NY, USA (1998)
5. Ho, E.S.L., Chan, J.C.P., Komura, T., Leung, H.: Interactive partner control in close interactions for real-time applications. ACM Trans. Multimedia Comput. Commun. Appl. **9**(3), 21:1–21:19 (Jul 2013)
6. Ho, E.S.L., Wang, H., Komura, T.: A multi-resolution approach for adapting close character interaction. In: Proceedings of the 20th ACM Symp. on Virtual Reality Software and Technology. pp. 97–106. VRST '14, ACM, New York, NY, USA (2014)
7. Ho, E.S., Komura, T., Tai, C.L.: Spatial relationship preserving character motion adaptation. ACM Transactions on Graphics (TOG) **29**(4), 33 (2010)
8. Hu, R., van Kaick, O., Wu, B., Huang, H., Shamir, A., Zhang, H.: Learning how objects function via co-analysis of interactions. ACM TOG **35**(4), 47:1–47:13 (Jul 2016)
9. Hu, R., Zhu, C., van Kaick, O., Liu, L., Shamir, A., Zhang, H.: Interaction context (icon): Towards a geometric functionality descriptor. ACM TOG **34**(4), 83:1–83:12 (Jul 2015)
10. Jin, T., Kim, M., Lee, S.H.: Aura mesh: Motion retargeting to preserve the spatial relationships between skinned characters. Comp. Graph. Forum **37**(2), 311–320 (2018)
11. Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.P.: Interactive multi-resolution modeling on arbitrary meshes. In: Proc. of the 25th Annual Conference on Comp. Graph. and Int. Tech. pp. 105–114. SIGGRAPH '98, ACM (1998)
12. Lee, J., Shin, S.Y.: A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of the 26th Annual Conference on Comp. Graph. and Inter. Tech. pp. 39–48. SIGGRAPH '99 (1999)
13. Müller, M., Gross, M.: Interactive virtual materials. In: Proceedings of Graphics Interface 2004. pp. 239–246. GI '04, Canadian Human-Comp. Comm. Soc. (2004)
14. Pirk, S., Krs, V., Hu, K., Rajasekaran, S.D., Kang, H., Yoshiyasu, Y., Benes, B., Guibas, L.J.: Understanding and exploiting object interaction landscapes. ACM Trans. Graph. **36**(3) (Jun 2017)
15. Si, H., Gärtner, K.: Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In: Proc. of the 14th Int. Meshing Roundtable. pp. 147–163 (2005)
16. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Proc. of the 5th EG Symp. on Geometry Processing. pp. 109–116. SGP '07, EG Assoc. (2007)
17. Tak, S., Ko, H.S.: A physically-based motion retargeting filter. ACM Trans. Graph. **24**(1), 98–117 (Jan 2005)
18. Villegas, R., Yang, J., Ceylan, D., Lee, H.: Neural kinematic networks for unsupervised motion retargetting. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
19. Zhao, X., Choi, M., Komura, T.: Character-object interaction retrieval using the interaction bisector surface. Comp. Graph. Forum **36**(2), 119–129 (May 2017)
20. Zhao, X., Wang, H., Komura, T.: Indexing 3d scenes using the interaction bisector surface. ACM Trans. Graph. **33**(3), 22:1–22:14 (Jun 2014)